

**KONRAD
LORENZ**
FUNDACIÓN UNIVERSITARIA





Electiva I: Backend para Web, Móvil e IoT

Tema 5

Profesor: Jorge Iván Andrés Contreras Pereira

Agenda



Agenda

- Javascript
- Typescript

Javascript



Lenguajes Estáticos vs. Dinámicos

Lenguajes estáticos

- Declaración de tipo explícita
- ```
string name = "Pesho"
int age = 25;
```
- Fuertemente tipado
- La verificación de tipos ocurre en tiempo de compilación
- Idiomas de tipo estático: C, C ++, C #, Java

## Lenguajes dinámicos (scripting)

- Declaración de tipo implícita
- ```
let name = "Pesho"
let age = 25;
```
- Débilmente tipado
- La verificación de tipos ocurre en tiempo de ejecución
- Lenguajes dinámicos:
JavaScript, PHP, Python, Ruby

Chrome Web Browser

Developer Console: [F12]

Elements

Console

Sources

top

☐ Preserve log

>

function add(a, b) {

return a + b;

}

<

undefined

>

add(5, 3)

<

8

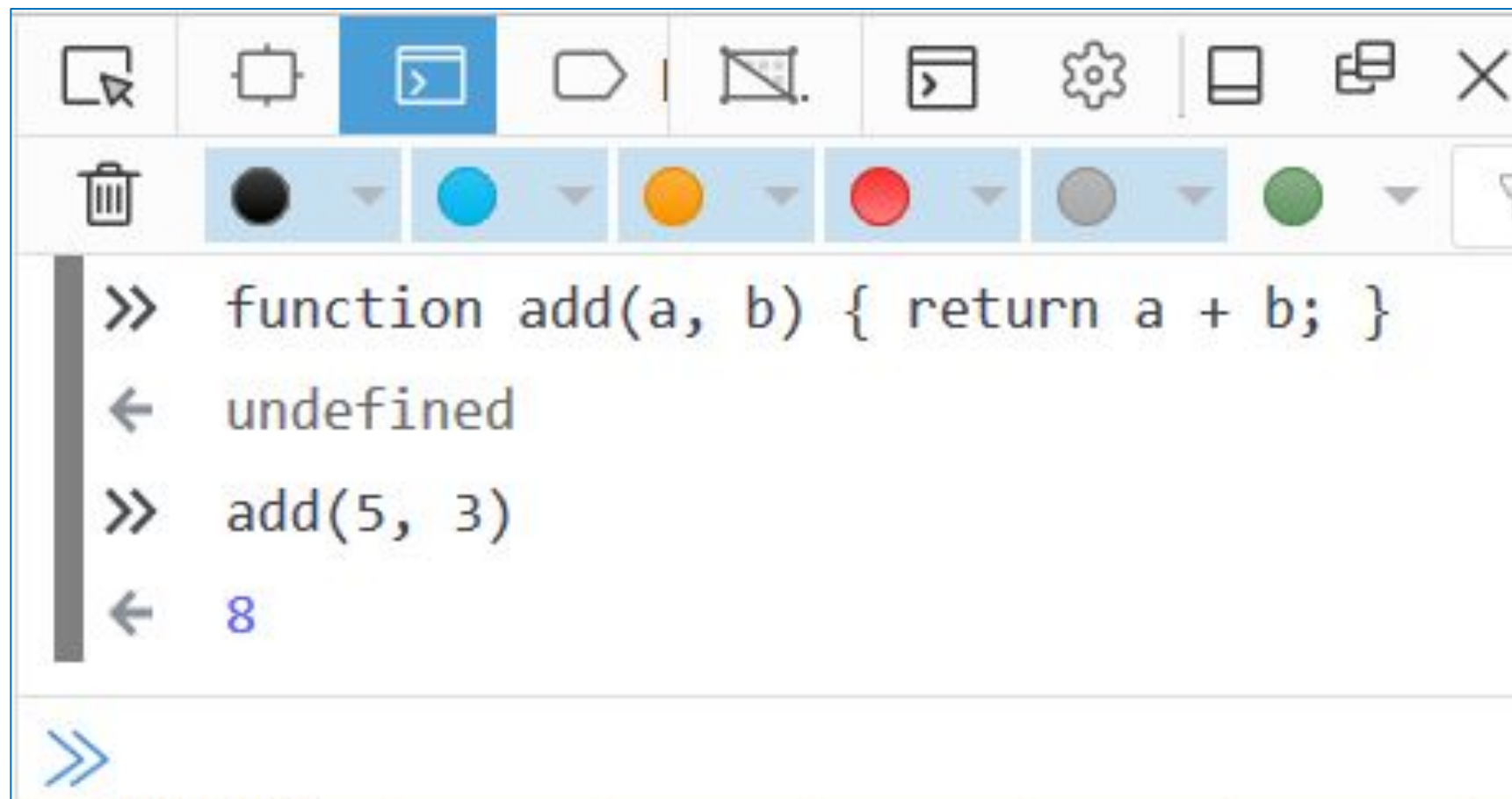
>

|



Firefox Web Browser

Developer Console: [Ctrl] + [Shift] + [i]



The screenshot shows the Firefox Developer Console interface. The top toolbar includes icons for various developer tools, with the 'Console' icon (a terminal window) highlighted in blue. Below the toolbar, the console area displays the following code and output:

```
>> function add(a, b) { return a + b; }
< undefined
>> add(5, 3)
< 8
```

At the bottom of the console, there is a blue prompt icon (>>) indicating where to enter new commands.



Instalar Node.js

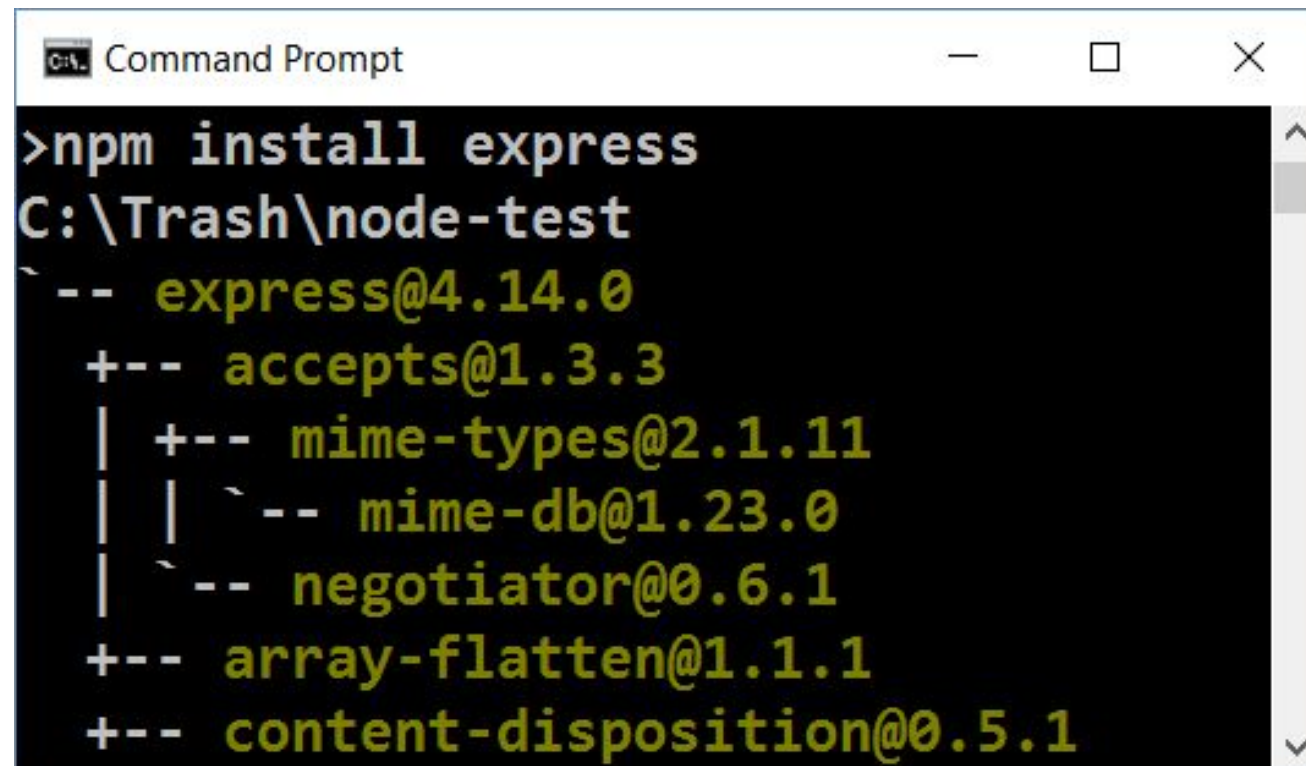


The screenshot shows the Node.js website with the following content:

- Navigation Bar:** HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS
- Introduction:** Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.
- Download for Windows (x64):**
 - v6.10.3 LTS** (Recommended For Most Users)
 - v7.10.0 Current** (Latest Features) - Highlighted with a yellow arrow and a red border.
- Footer Links:** Other Downloads | Changelog | API Docs (repeated for both versions)

¿Qué es Node.js?

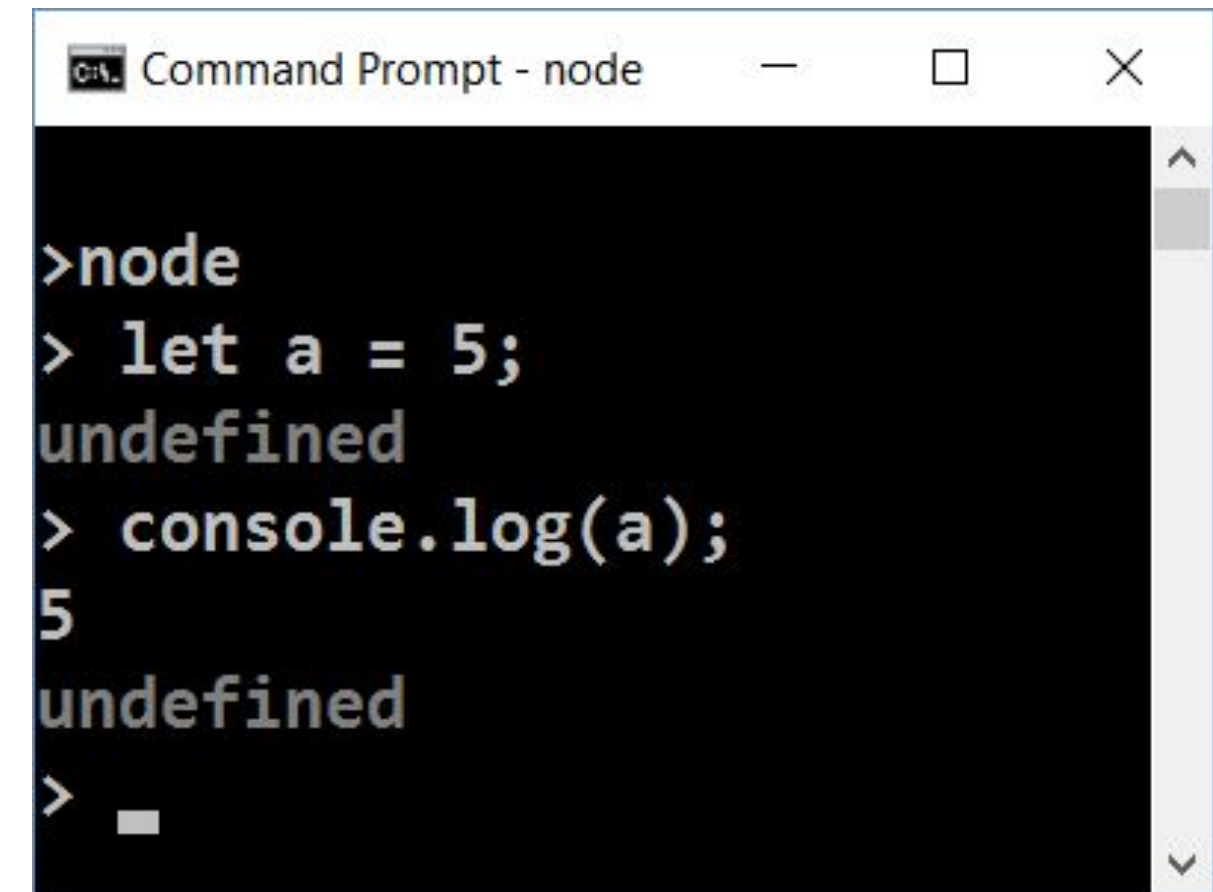
- Tiempo de ejecución de **JavaScript** del lado del servidor
- Motor de JavaScript Chrome **V8**



```

>npm install express
C:\Trash\node-test
`-- express@4.14.0
   +-- accepts@1.3.3
   |   +-- mime-types@2.1.11
   |   |   `-- mime-db@1.23.0
   |   `-- negotiator@0.6.1
   +-- array-flatten@1.1.1
   +-- content-disposition@0.5.1

```

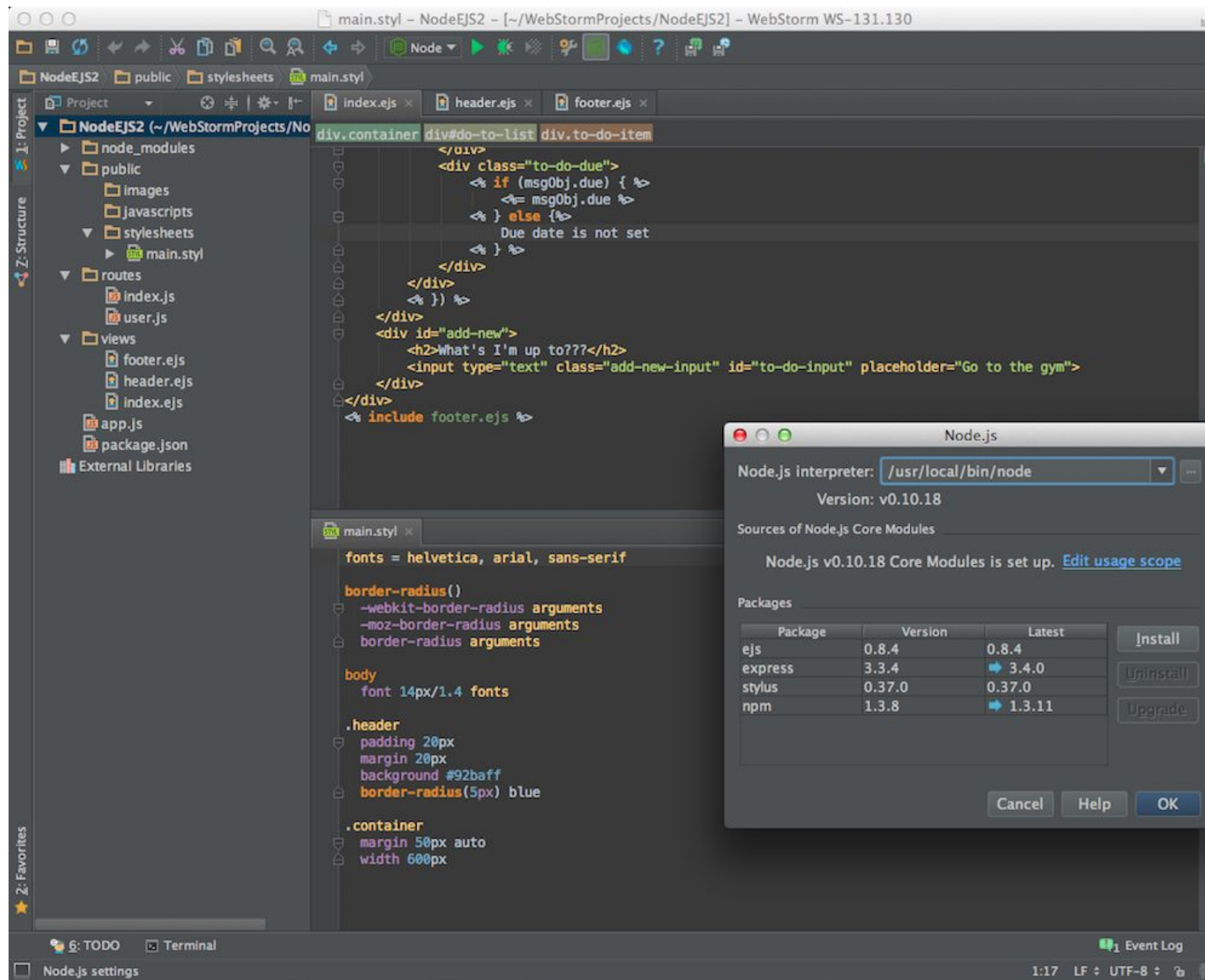


```

>node
> let a = 5;
undefined
> console.log(a);
5
undefined
>

```

- **npm** package manager
 - Instalador de paquetes de node



**Recomendación
WebStorm**

IDE muy inteligente
de JavaScript

Sintaxis de Javascript



Sintaxis JavaScript

- La sintaxis de JavaScript es similar a **C#, Java y PHP**
- Operadores (+, *, =, !=, &&, ++, ...)
- Variables (sin tipo)
- Declaraciones condicionales (**if, else**)
- Bucles (**for, while**)
- Funciones (con parámetros y valor de retorno -return-)
- Arrays (arr [5]) y arrays asociativos (arr ['maria'])
- Objetos y clases

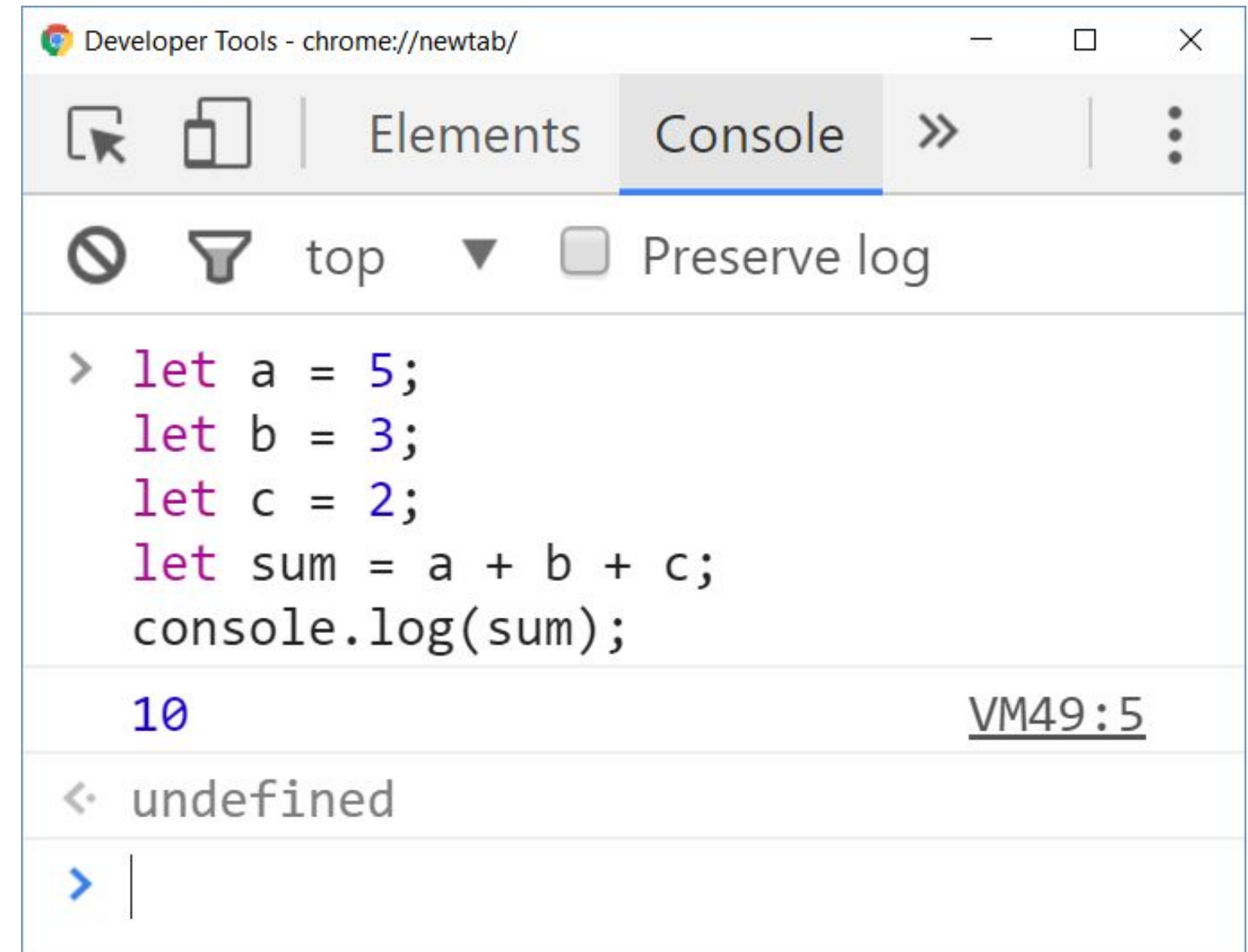


Código JavaScript - Ejemplo

- JS variables, operators, expressions, statements

```
let a = 5;
let b = 3;
let c = 2;

let sum = a + b + c;
console.log(sum);
```



Función para sumar dos números

Uso `camelCase` para nombre de funciones

El `{` permanece en la misma línea

```
function sumNumbers(a, b, c) {
```

```
    let sum = a + b + c;
```

La entrada viene como argumentos numéricos

```
    console.log(sum);
```

Imprime la suma en la consola

```
}
```

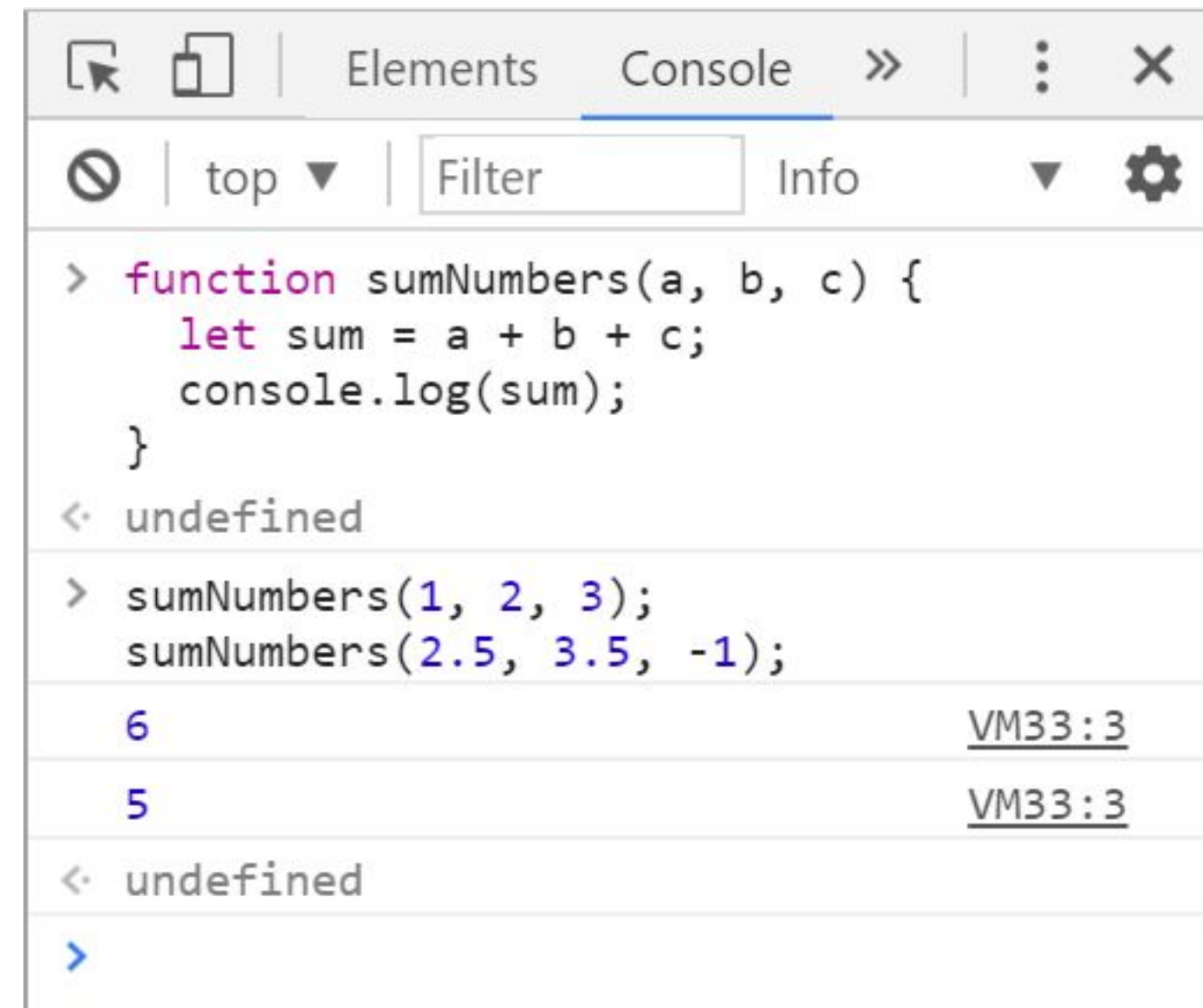
Probando en el navegador

- Llame a la función con tres argumentos pasando los números:

```
sumNumbers(1, 2, 3);  
// 6
```

```
sumNumbers(2.5, 3.5, -1);  
// 5
```

```
sumNumbers(-1, -2, -3);  
// -6
```



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following code and output:

```
> function sumNumbers(a, b, c) {  
    let sum = a + b + c;  
    console.log(sum);  
}  
< undefined  
  
> sumNumbers(1, 2, 3);  
    sumNumbers(2.5, 3.5, -1);  
  
6 VM33:3  
5 VM33:3  
< undefined  
>
```


Tipos de datos



Números en Javascript

- Todos los números en JS son de coma flotante (64 bits de precisión doble)

```
let n = 5;
```

```
console.log(typeof(n) + ' ' + n); // number 5
```

```
let bin = 0b10000001;
```

```
console.log(typeof(bin) + ' ' + bin); // number 129
```

```
let x = Number.parseInt("2.99");
```

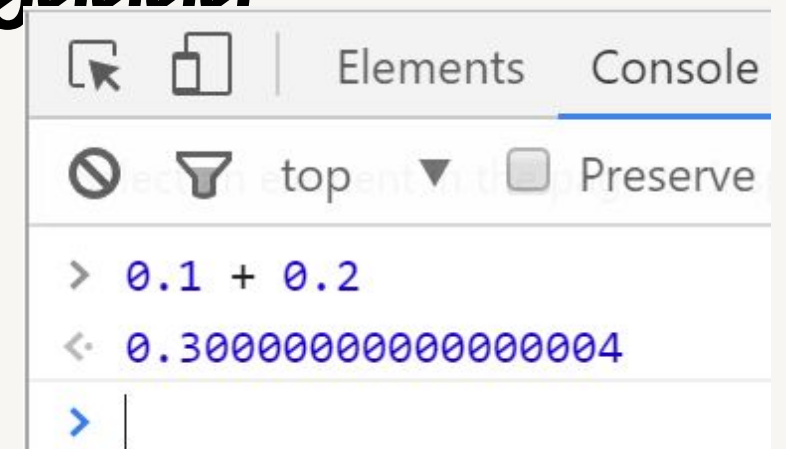
```
console.log(typeof(x) + ' ' + x); // number 2
```

```
let y = Number("2.99");
```

```
console.log(typeof(y) + ' ' + y); // number 2.99
```

Los números son de punto flotante

```
let a = 3, b = 4 / a;
console.log(`a=${a} b=${b}`); // a=3 b=1.3333333333333333
console.log(a / 0); // Infinity
console.log(a * Infinity); // Infinity
console.log(0 / 0); // NaN
let bigNum = 1000000000000000000000000;
console.log(bigNum + 1); // 1000000000000000000000000
let num = 0.01, sum = 0;
for (let i=0; i<100; i++)
    sum += num;
console.log(sum); // 1.000000000000000000000007
```



Strings in JavaScript

- Las cadenas en JS son secuencias inmutables de caracteres Unicode

```
let ch = 'x';
console.log(typeof(ch) + ' ' + ch); // string x

let str = "hola";
console.log(typeof(str) + ' ' + str); // string hola
console.log(str.length); // 5
console.log(str[0]); // h
str[0] = 's'; // Cuidado: no hay error, ¡pero str permanece sin cambios!
console.log(str); // hola
console.log(str[10]); // undefined
```

En modo estricto esto da error

Objetos en Javascript

- Los objetos JS tienen pares clave-valor

```
let point = {x: 5, y: -2};
console.log(point); // Object {x: 5, y: -2}
console.log(typeof(point)); // object

let rect = {name:"habitación",width: 4.5,height: 3.5};
rect.name = "habitación";
rect.color = "amarilla";
console.log(rect); // Object {name: "habitación", width: 4.5,
height: 3.5, color: "amarilla"}
rect = undefined; // "rect" ahora no tiene valor
console.log(rect); // undefined
```

Asignación de variables

- Las variables en JS no tienen tipo, los valores tienen tipo

```
let x = 5; console.log(typeof(x)); // x es número  
x = 'str'; console.log(typeof(x)); // x ahora es string  
x = {x:5, y:7}; console.log(typeof(x)); // x ahora es objeto
```

```
let n = 5, str = "hello"; // Asigna valores
```

Typescript



Conceptos básicos - clases



Conceptos básicos de POO - Clase

En informática, una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos. Están compuestas por métodos, y atributos.

Conceptos básicos de POO – Clase

Declaración en Typescript

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting;
  }
}

let greeter = new Greeter("world");
```

Atributo



Conceptos básicos de POO - Atributo

Contenedor de información de una clase. Una clase puede tener uno o varios atributos.

Conceptos básicos de POO - Atributo

```
class clase{
    unAtributo : string;
    otroAtributo : boolean;
    unAtributoMas : number;

    constructor(argument) {
        // code...
    }
}
```

Método



Conceptos básicos de POO – Método

Secuencia de instrucciones de que pertenecen a una una clase o a un objeto, y que es ejecutada al recibir un mensaje.

Conceptos básicos de POO – Método

```
class Greeter {
    greeting: string;

    constructor(message: string) {
        this.greeting = message;
    }
    unMetodo() {
        return "Hello, " + this.greeting
        ;
    }
}
```


Objeto



Conceptos básicos de POO – Objeto

Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

Conceptos básicos de POO – Objeto

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting
  }
}

let nuevoObjeto = new Greeter("world");
```

Herencia



Conceptos básicos de POO – Herencia

La herencia es un mecanismo de los lenguajes de programación orientada a objetos basados en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad. La clase de la que se hereda se suele denominar clase base, clase padre, superclase, clase ancestro.

Conceptos básicos de POO – Herencia

Clase padre

```
class Person {  
    protected name: string;  
    protected constructor(theName:  
        string) { this.name = theName; }  
}
```

Conceptos básicos de POO – Herencia

Clase hija

```
// Employee can extend Person
class Employee extends Person {
    private department: string;

    constructor(name: string, department
        : string) {
        super(name);
        this.department = department;
    }

    public getElevatorPitch() {
        return `Hello, my name is ${this
            .name} and I work in ${this.
            department}.`;
    }
}
```

Conceptos de Typescript



Introducción a TypeScript - ¿Qué es TS?

“TypeScript is a typed superset of JavaScript that compiles to plain JavaScript”

[Presentación TypeScript – video](#)

Introducción a TypeScript - ¿Qué es TS?

TypeScript inicia desde la misma sintaxis y semántica de JavaScript que millones de desarrolladores conocen:

```
// TypeScript
function unaFuncion( mensaje : string) {
    console.log("El mensaje es: " + mensaje);
}
```

```
// JavaScript
function unaFuncion(mensaje) {
    console.log("El mensaje es: " + mensaje);
}
```

Declaración de variables



Introducción a TypeScript – Declaración de variables

TypeScript nos permite declarar variables del mismo modo que se hace en JavaScript, con la ventaja de poder especificar el tipo de dato de la misma.

```
// TypeScript
//Boolean
let isDone: boolean = false;
//Number
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
```

```
// JavaScript
//Boolean
var isDone = false;
//Number
var decimal = 6;
var hex = 0xf00d;
var binary = 10;
var octal = 484;
```


Introducción a TypeScript – Declaración de variables

Junto con los anteriores, estos son algunos de los tipos de variables nativas de TypeScript:

```
// TypeScript
//String
let color: string = "blue";
color = 'red';
let fullName: string = `Bob Bobbington`;
let age: number = 37;
// Arrays:
let list: Array<number> = [1, 2, 3];
```

```
// JavaScript
//String
var color = "blue";
color = 'red';
var fullName = "Bob Bobbington";
var age = 37;
// Arrays:
var list = [1, 2, 3];
```

Funciones



Introducción a TypeScript - Funciones

Volviendo al ejemplo anterior, observamos como la declaración de funciones es prácticamente igual a como se realizan en JavaScript.

```
// TypeScript
function unaFuncion( mensaje : string) {
    console.log("El mensaje es: " + mensaje);
}
```

```
// JavaScript
function unaFuncion(mensaje) {
    console.log("El mensaje es: " + mensaje);
}
```

Classes



Introducción a TypeScript - Clases

TypeScript nos permite declarar clases de forma natural, de forma similar como lo haríamos en otros lenguajes orientados a objetos:

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting;
  }
}

let greeter = new Greeter("world");
```

Introducción a TypeScript - Clases

Sin embargo vemos una clara diferencia con respecto a su traducción en JavaScript

```
var Greeter = /** @class */ (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        return "Hello, " + this.greeting;
    };
    return Greeter;
})();
```

Interfaces



Introducción a TypeScript – Interfaces

TypeScript también nos brinda una forma sencilla de declarar interfaces:

```
// TypeScript
interface LabelledValue {
    label: string;
}
// Valores opcionales:
interface SquareConfig {
    color?: string;
    width?: number;
}
```



Gracias